Prediction and Causal Reasoning in Planning

T. Dean and M. Boddy
Brown University
Providence, RI 02912

Abstract

Nonlinear planners (e.g., NOAH [10]) are often touted as having an efficiency advantage over linear planners (e.g., STRIPS [8]). The reason usually given is that nonlinear planners, unlike their linear counterparts, are not forced to make arbitrary commitments to the order in which actions are to be performed. This ability to delay commitment enables nonlinear planners to solve certain problems with far less effort than would be required of linear planners. In this paper, we argue that this advantage is bought with a significant reduction in the ability of a nonlinear planner to accurately predict the consequences of actions. Unfortunately, the general problem of predicting the consequences of a partially ordered set of actions is intractable. In gaining the predictive power of linear planners, nonlinear planners sacrifice their efficiency advantage. There are, however, other advantages to nonlinear planning (e.g., the ability to reason about partial orders and incomplete information) that make it well worth the effort needed to extend nonlinear methods. In this paper, we supply a framework for causal inference that supports reasoning about partially ordered events and actions whose effects depend upon the context in which they are executed. As an alternative to a complete but potentially exponential-time algorithm, we provide a provably sound polynomial-time algorithm for predicting the consequences of partially ordered events. If the events turn out to be totally ordered, then the algorithm is complete as well as sound.

Keywords: causal reasoning, planning, prediction, nonmonotonic inference, data base management.

1 Introduction

In this paper, we are concerned with the process of incrementally constructing *nonlinear* plans (i.e., plans represented as sets of actions whose order is only partially specified). Nonlinear planning [2] has long been considered to have distinct advantages over linear planning systems such as STRIPS [8] and its descendents. One supposed advantage [10] has to do with the idea that, by delaying commitment to the order in which independent actions are to be performed, a planner can avoid unnecessary backtracking. Linear planners are often forced to make arbitrary commitments regarding the order in which actions are to be carried out. Such arbitrary orderings often fail to lead to a solution and have to be reversed. By ordering only actions known to interact with one another (i.e., actions whose outcomes depend upon the order in which they are executed) the expectation was that nonlinear planners would avoid a lot of unnecessary work.

The problem in getting delayed-commitment planning to work is that it is often difficult to determine if two actions actually are independent. In order to determine whether or not two actions are independent, it is necessary to determine what the effects of those actions are. Unfortunately, in order to determine the

effects of an action a it is necessary to determine what is true prior to a being executed, and this in turn requires that we know the effects of those actions that precede a. In general there is no way to determine whether or not two actions are independent without actually considering all of the possible total orderings involving those two actions.

Planning depends upon the ability to predict the consequences of acting. Past planning systems capable of reasoning about partial orders (i.e., nonlinear planners) have either employed weak (and often unsound) methods for performing predictive inference or they have sought to delay prediction until the conditions immediately preceding an action are known with certainty. Delaying predictive inference can serve to avoid inconsistency, but it can also result in extensive backtracking in those very situations that nonlinear planners were designed to handle efficiently.

It our contention that the initial success of Sacerdoti's NOAH [10] program and the promise of NOAH's style of least-commitment planning has caused researchers to ignore important issues in reasoning with incomplete information. The idea of least-commitment planning is not the only reason for building planners capable of reasoning about partially ordered events. Most events will not be under a planner's control and more often than not it will impossible to determine the order of all events with absolute certainty. Planning systems for realistic applications will have to reason about partially ordered events.

In this paper, we consider the problem of reasoning about the effects of partially ordered actions. A theory for reasoning about the effects of actions (or, more generally, the consequences of events) is referred to as a *causal theory*. We will describe a language for constructing causal theories that is capable of representing indirect effects and the effects of actions that depend upon the situation in which they are applied. We will describe a series of algorithms for reasoning about such causal theories. All of these algorithms are polynomial-time, incremental, and insensitive to the order in which facts are added to or deleted from the data base. We show that a particular algorithm is complete for causal theories in which the events are totally ordered, but is potentially inconsistent in cases where the events are not totally ordered. In [6] we show that the general problem of reasoning about conditional actions is *NP-complete*, and, in this paper, we provide a partial decision procedure that, while not complete, is provably sound. What this means for a planner constructing a plan is that the procedure is guaranteed not to misslead the planner into committing to a plan that is provably impossible given what is currently known. If the decision procedure answers yes, then the conditions are guaranteed to hold in every totally ordered extension of the current partial order; if the decision procedure answers no, there is a chance that the conditions hold in every total order, but to determine this with certainty might require an exponential amount of time or space.

2 Temporal Data Base Management

In this section, we consider a particular type of inference system, referred to as a *temporal data base management system* (or TDBMS) [4], that is used to keep track of what is known about the order, duration, and time of occurrence of a set of events and their consequences. The user of a TDBMS is allowed to add two sorts of information: that which is unconditionally believed and that which is believed just in case certain conditions can be shown to hold. The former includes information concerning events that have

been observed or are assumed inevitable and information in the form of general rules that are believed to govern the physics of a particular domain.

In specifying conditional beliefs, the user explicitly states what the conditions are, and the TDBMS ensures that those beliefs (and their consequences) are present in the data base just in case the conditions are met. This is achieved through the use of *data dependencies* [7]. In a TDBMS, the primary forms of data dependency (in addition to those common in static situations) are concerned with some fact being true at a point in time or throughout an interval. In addition, there is a nonmonotonic form of temporal data dependency concerned with it being *consistent* to believe that a fact is not true at a point in time or during any part of an interval. These forms of temporal data dependency are handled in the TDBMS using the mechanism of *temporal reason maintenance* [4]. Language constructs are supplied in the TDBMS that allow an application program to query the data base in order to establish certain antecedent conditions (including temporal conditions) and then, on the basis of these conditions, to assert consequent predictions. These predictions remain valid just in case the antecedent conditions continue to hold.

Perhaps one of the most important and most overlooked characteristics of a temporal reasoning system is the ability to handle incomplete information of the sort one invariably encounters in realistic applications. For example, we seldom know the exact duration or time of occurrence of most events. Moreover, for those durations and offsets we do know, they are seldom with respect to a global frame of reference such as a clock or calendar. In the TDBMS, every point is a frame of reference, and it is possible to constrain the distance between any two points simply by specifying bounds on the distance in time separating the two points. By allowing bounds to be both numeric and symbolic, the same framework supports both qualitative (i.e. ordering) and quantitative (distance) relationships.

Another important aspect of reasoning with incomplete information has to do with the default character of temporal inference. In general, it is difficult to predict in advance how long a fact made true will persist. It would be convenient to leave it up to the system to decide how long facts persist based upon the simple default rule [9] that a fact made true continues to be so until something serves to make it false. This is exactly what the TDBMS does. The term *persistence* is used to refer to an interval corresponding to a particular (type of) fact becoming true and remaining so for some length of time. A fact is determined to be true throughout an interval I just in case there is a persistence that begins before the beginning of I and it can't be shown that the persistence ends before the end of I.

The TDBMS permits the specification of partial orders, but it imposes orderings in situations leading to potential incoherency. If the TDBMS encounters two persistence intervals of contradictory types that are not ordered with respect to one another, it prompts the calling program to resolve the possible contradiction by either imposing an order or explicitly introducing a disjunction. By introducing a disjunction, the calling program effectively splits the data base, producing two time lines. The answers returned by queries to the TDBMS indicate the disjuncts that must be true for a query to succeed (i.e., the particular time line that satisfies the query). There are also language constructs that allow a calling program to eliminate disjuncts (and hence time lines) that have been ruled out. Unfortunately, as we will see in Section 5, eliminating explicit contradictions is not sufficient to ensure consistency in a system capable of making conditional predictions from a set of partially ordered events [1]. Before we continue our discussion it will help to introduce some notation.

Relations. Let Π be the set of points corresponding to the begin and end of events in a particular temporal data base. We define a function DIST to denote the best known bounds on the distance in time separating two points. Given $\pi_1, \pi_2 \in \Pi$ such that $\text{DIST}(\pi_1, \pi_2) = \langle low, high \rangle$, we have:

- $\pi_1 \prec \pi_2 \Leftrightarrow low \geq \epsilon$[1] — π_1 precedes π_2
- $\pi_1 \equiv \pi_2 \Leftrightarrow \langle low, high \rangle = \langle 0, 0 \rangle$ — π_1 is coincident with π_2
- $\pi_1 \preceq \pi_2 \Leftrightarrow (\pi_1 \prec \pi_2) \vee (\pi_1 \equiv \pi_2)$ — π_1 precedes or is coincident with π_2
- $\pi_1 \prec_M \pi_2 \Leftrightarrow high \geq \epsilon$ — π_1 possibly precedes π_2
- $\pi_1 \preceq_M \pi_2 \Leftrightarrow high \geq 0$ — π_1 possibly precedes or is coincident with π_2

Tokens. We denote a set of *time tokens* $\mathsf{T} = \{t_0, t_1, \ldots t_n\}$ for referring to intervals of time during which certain events occur or certain facts are known to become true and remain so for some period of time. The latter correspond to what we have been calling persistences. For a given token t:

- BEGIN(t), END(t) $\in \Pi$.
- STATUS(t) $\in$ {IN,OUT}, determined by whether the token is warranted (IN) or not (OUT) by the current premises and causal theory.
- TYPE(t) = P where P is an atomic predicate calculus formula with no variables
- DURATION(t) = DIST(BEGIN(t), END(t))

Types. As defined above, the type of an individual token is an atomic formula with no variables (e.g., `(on block14 table42)`). In general, any atomic formula, including those containing variables, can be used to specify a type. In describing the user interface, universally quantified variables are notated `?variable-name`, the scope of the variable being the entire formula in which it is contained (e.g., `(on ?x ?y)`). In describing the behavior of the inference system, we will use variables of the form t_P to quantify over tokens of type P (e.g., $\forall t_P \in \mathsf{T}$ TYPE(t_P) = P).

3 Reasoning about Causality

In the TDBMS, a causal theory is simply a collection of rules, called *projection rules*, that are used to specify the behavior of processes. In the following rule, $P_1 \ldots P_n$, $Q_1 \ldots Q_m$, E, and R designate types, and *delay* and *duration* designate constraints (e.g., $\langle \epsilon, \infty \rangle$). In:

```
(project (and P1 ... Pn
              (M (not (and Q1 ... Qm))))
         E  delay  R  duration)                    (1)
```

$P_1 \ldots P_n$ and $Q_1 \ldots Q_m$ are referred to as *antecedent conditions*, E is the type of the *triggering event*, and R refers to the type of the *consequent prediction*. The above projection rule states that, if an event

[1] The symbol ϵ is meant to denote an infinitesimal: a number greater than 0 and smaller than any positive number.

R1: (project (and $P_1 \ldots P_n$
(M (not (and $Q_1 \ldots Q_m$))))
E R)

R2: (project (and $P_1 \ldots P_n$)
E R)

R3: (disable (and $Q_1 \ldots Q_m$)
(ab R2))

R4: (disable (and $R_1 \ldots R_o$)
(ab R3))

Figure 1: Hierarchically arranged projection and disabling rules

of type E occurs corresponding to the token t_E and $P_1 \ldots P_n$ are believed to be true at the outset[2] of t_E and it is consistent to believe that the conjunction of $Q_1 \ldots Q_m$ is not true at the outset of t_E, then, after an interval of time following the end of t_E determined by *delay*, R will become true and remain so for a period of time constrained by *duration* (if *delay* and *duration* are not specified, they default to $\langle 0, 0 \rangle$ and $\langle \epsilon, \infty \rangle$, respectively). In the following, we will be considering a restricted form of causal theory, called a *type 1* theory, such that the *delay* always specifies a positive offset (causes always precede their effects).

We also allow the user to specify rules that serve to *disable* other rules [11]. Figure 1 shows a standard projection rule R1 and a pair of projection and disabling rules R2 and R3 that replace R1. The rule R3 is further conditioned by the rule R4. Assuming just the rules R2, R3, and R4, any application of R2 with respect to a particular token t of type E is said to be abnormal with regard to t just in case $Q_1 \ldots Q_m$ hold at the outset of t and it is consistent to believe that R3 is not abnormal with regard to t. The nonmonotonic behavior of type 1 causal theories is specified entirely in terms of disabling rules and the default rule of persistence (see Section 2). In addition to their usefulness for handling various forms of incomplete information, disabling rules make it possible to reason about the consequences of simultaneous actions. The reader interested in a more detailed treatment of causal theories may refer to one of [4], [5], or [11]. In parts of this paper, we will ignore disabling rules and speak of causal theories consisting solely of *simple projection rules* of the form (project (and $P_1 \ldots P_n$) E R).

One of the most problematic aspects of designing a temporal inference system involves defining precisely what it *means* for a fact to be true at a point or throughout an interval (i.e., the conditions under which a query of the form $TT(P, \pi_1, \pi_2)$ will succeed). As a first approximation, we offer the following definition, which we will refer to as *weak true throughout*:

[2] An alternative formulation described in [3] states that the antecedent conditions of a projection rule must be true *throughout* the trigger event rather than true just at the outset. Both formulations are supported in the TDBMS, though we will only be discussing the true-at-the-outset formulation in this paper.

$$
\begin{array}{l}
\forall t_E \in \mathsf{T} \\
\quad ((\textsc{status}(t_E) = \textsc{in}) \wedge \\
\quad (\exists t_{P_1} \ldots t_{P_n} \in \mathsf{T} \\
\quad\quad (\forall\, 1 \le i \le n\ (\textsc{status}(t_{P_i}) = \textsc{in}) \wedge \\
\quad\quad\quad\quad (\textsc{begin}(t_{P_i}) \preceq \textsc{begin}(t_E)) \wedge \\
\quad\quad\quad\quad (\textsc{begin}(t_E) \preceq_M \textsc{end}(t_{P_i}))\ )) \wedge \\
\quad\quad \neg(\exists t_{Q_1} \ldots t_{Q_m} \in \mathsf{T} \\
\quad\quad\quad (\forall\, 1 \le j \le m\ (\textsc{status}(t_{Q_j}) = \textsc{in}) \wedge \\
\quad\quad\quad\quad\quad (\textsc{begin}(t_{Q_j}) \preceq \textsc{begin}(t_E)) \wedge \\
\quad\quad\quad\quad\quad (\textsc{begin}(t_E) \preceq_M \textsc{end}(t_{Q_j}))\ ))) \\
\quad\quad \Rightarrow\ \exists t_R \in \mathsf{T} \\
\quad\quad\quad (\textsc{status}(t_R) = \textsc{in}) \wedge \\
\quad\quad\quad (\textsc{dist}(\textsc{end}(t_E), \textsc{begin}(t_R)) \subseteq \mathit{delay}) \wedge \\
\quad\quad\quad (\textsc{dist}(\textsc{begin}(t_R), \textsc{end}(t_R)) \subseteq \mathit{duration})
\end{array}
$$

Figure 2: Weak projection

$$
\begin{array}{l}
\forall \pi_1 \pi_2 \in \Pi \\
\quad \exists t_P \in \mathsf{T} \\
\quad\quad (\textsc{status}(t_P) = \textsc{in}) \wedge \\
\quad\quad (\textsc{begin}(t_P) \preceq \pi_1) \wedge \\
\quad\quad (\pi_2 \preceq_M \textsc{end}(t_P)) \\
\quad \Leftrightarrow\ TT(P, \pi_1, \pi_2)
\end{array}
\qquad (2)
$$

In order to specify the behavior of a temporal inference system such as the TDBMS, we also need to define the criterion for inferring consequent effects from antecedent conditions via causal rules. Our first such criterion will be referred to as *weak projection* (Figure 2) and is defined with respect to the general form of a projection rule (1). Weak true throughout and weak projection correspond to the assumption that "what you don't know won't hurt you." Only those events that can be shown to be ordered with respect to a particular point will have any effect at that point. As we will see in Section 5, there are some problems with this formulation.

4 Transactions on the Data Base: the User Interface

Every inference system provides some means for the user to specify rules (referred to collectively as a *causal theory*) for inferring additional consequences of the data (referred to here as a set of *basic facts*). An application program interacts with an inference system by adding and removing items from the set of basic facts, which in the TDBMS corresponds to a set of tokens and a set of constraints. The state of a temporal data base is completely defined by a temporal constraint graph (TCG), consisting of the begin and end points of tokens and constraints between them, and a causal dependency graph (CDG), consisting of dependency structures corresponding to the application of causal rules in deriving new tokens. Each transaction performed on the temporal data base results in changes to these two data structures. The TDBMS is responsible for maintaining the temporal data base so that it captures exactly those consequences that follow from the causal theory and the current set of basic facts.

Generally, the causal theory remains fixed for a particular application, and interaction with the TDBMS consists of a series of transactions and queries. A transaction consists of either adding or removing some token or constraint from the set of basic facts. A query consists of a predicate calculus formula corresponding to a question of the form "Could some fact P be true over a particular interval I?" An affirmative answer returned by the TDBMS in response to such a query will include a set of assumptions necessary for concluding that the fact is indeed true. Any assertions made on the basis of the answer to such a query are made to depend upon these assumptions. There is also a mechanism that enables the TDBMS to detect and, with the assistance of the calling program, resolve inconsistencies in the set of constraints.

5 Completeness and Consistency

The primary source of ambiguity in the TDBMS arises from the fact that the set of constraints seldom determines a total ordering of the tokens in T. Given that most inferences depend only upon what is true during intervals defined by points corresponding to the begin and end of tokens in T, all that we are really interested in is what facts are true in what intervals in the different total orderings of time points consistent with the initial set of constraints. For each total ordering we can identify a unique set of tokens that intuitively should be IN given a particular causal theory.

As far as we are concerned, an *inference procedure* is fully specified by a criterion for inferring consequent effects from antecedent causes via causal rules (e.g. weak projection), a method for actually applying that criterion (an update algorithm), and a criterion for determining if a fact is true throughout some interval (e.g., weak true throughout). We will say that a particular inference procedure is *complete* for a class of causal theories, if for any set of basic facts and causal theory in the class, the statements of the form $TT(P, \pi_1, \pi_2)$ warranted by the inference procedure are exactly those that are true in all total orders consistent with the constraints in the TCG. Similarly, we will say that an inference procedure is *sound* for a class of causal theories, if for any set of basic facts and causal theory in the class, each statement $TT(P, \pi_1, \pi_2)$ warranted by the inference procedure is true for any total order consistent with the set of constraints. Given the preceding definitions, it is easy to show that the TDBMS is complete and sound for type 1 causal theories, assuming that the tokens in T are totally ordered [6].

In situations where the set of basic facts does not determine a total order, it's easy to show that the TDBMS can end up in a state with IN tokens that allow one to conclude statements of the form $TT(P, \pi_1, \pi_2)$ that are not true in *any* totally ordered extension. One thing we might do to improve the chances of the TDBMS warranting only valid statements of the form $TT(P, \pi_1, \pi_2)$ is strengthen the criterion for belief in a given token. We can determine a class of tokens that are said to be *strongly protected*, using the axioms in Figure 3. In these axioms and the rest of the paper, we use T_B to denote the tokens in the set of basic facts. If the set of constraints determines a total ordering, then the set of strongly protected tokens is identical to the set of tokens that are IN, but generally the former is a subset of the latter. Using this notion of strongly protected, we can define a stronger true throughout criterion that we will refer to as *strong true throughout*:

$$
\begin{array}{l}
\forall t \in T_B \\
\quad \textit{STRONGLY-PROTECTED}(t) \\
\\
\forall t_B \in T \\
\quad (\textit{STRONGLY-PROTECTED}(t_B) \wedge \\
\quad (\exists t_{P_1} \ldots t_{P_n} \in T \\
\quad\quad (\forall\, 1 \leq i \leq n\ \textit{STRONGLY-PROTECTED}(t_{P_i}) \wedge \\
\quad\quad\quad\quad (\textsc{begin}(t_{P_i}) \preceq \textsc{begin}(t_B)) \wedge \\
\quad\quad\quad\quad (\textsc{begin}(t_B) \preceq_M \textsc{end}(t_{P_i})) \wedge \\
\quad\quad\quad\quad (\forall t_{\neg P_i} \in T \\
\quad\quad\quad\quad\quad\quad (\textsc{status}(t_{\neg P_i}) = \textsc{out}) \vee \\
\quad\quad\quad\quad\quad\quad (\textsc{begin}(t_{\neg P_i}) \prec \textsc{begin}(t_{P_i})) \vee \\
\quad\quad\quad\quad\quad\quad (\textsc{end}(t_B) \prec \textsc{begin}(t_{\neg P_i}))))))) \\
\quad \Rightarrow \exists t_R \in T \\
\quad\quad \textit{STRONGLY-PROTECTED}(t_R) \wedge \\
\quad\quad (\textsc{dist}(\textsc{end}(t_B), \textsc{begin}(t_R)) \subseteq \textit{delay}) \wedge \\
\quad\quad (\textsc{dist}(\textsc{begin}(t_R), \textsc{end}(t_R)) \subseteq \textit{duration})
\end{array}
$$

Figure 3: Strong protection defined for simple projection rules

$$
\begin{array}{l}
\forall t_B \in T \\
\quad ((\textsc{status}(t_B) = \textsc{in}) \wedge \\
\quad (\exists t_{P_1} \ldots t_{P_n} \in T \\
\quad\quad (\forall\, 1 \leq i \leq n\ (\textsc{status}(t_{P_i}) = \textsc{in}) \wedge \\
\quad\quad\quad\quad (\textsc{begin}(t_{P_i}) \preceq_M \textsc{begin}(t_B)) \wedge \\
\quad\quad\quad\quad (\textsc{begin}(t_B) \preceq_M \textsc{end}(t_{P_i})) \wedge \\
\quad\quad\quad\quad (\forall t_{\neg P_i} \in T \\
\quad\quad\quad\quad\quad\quad \neg \textit{STRONGLY-PROTECTED}(t_{\neg P_i}) \vee \\
\quad\quad\quad\quad\quad\quad (\textsc{begin}(t_{\neg P_i}) \prec_M \textsc{begin}(t_{P_i})) \vee \\
\quad\quad\quad\quad\quad\quad (\textsc{begin}(t_B) \prec_M \textsc{begin}(t_{\neg P_i}))))))) \\
\quad \Rightarrow \exists t_R \in T \\
\quad\quad (\textsc{status}(t_R) = \textsc{in}) \wedge \\
\quad\quad (\textsc{dist}(\textsc{end}(t_B), \textsc{begin}(t_R)) \subseteq \textit{delay}) \wedge \\
\quad\quad (\textsc{dist}(\textsc{begin}(t_R), \textsc{end}(t_R)) \subseteq \textit{duration})
\end{array}
$$

Figure 4: Improbably weak projection defined for simple projection rules

$$
\begin{array}{l}
\forall \pi_1 \pi_2 \in \Pi \\
\quad \exists t_P \in T \\
\quad\quad \textit{STRONGLY-PROTECTED}(t_P) \wedge \\
\quad\quad (\textsc{begin}(t_P) \preceq \pi_1) \wedge \\
\quad\quad (\pi_2 \preceq_M \textsc{end}(t_P)) \\
\quad \Leftrightarrow TT(P, \pi_1, \pi_2)
\end{array}
\qquad (3)
$$

As it turns out, weak projection and strong true throughout still do not constitute a sound inference procedure. We can show that, even when we restrict ourselves to strongly protected tokens, most interesting decision problems are intractable. In fact, we can prove that the problem of determining if $TT(P, \pi_1, \pi_2)$ is

true for a type 1 causal theory, with or without disabling rules, is *NP-complete* [6]. Although completeness is computationally infeasible, it is possible to devise an inference procedure that is both sound and capable of performing useful prediction. First, we provide a criterion for generating consequent predictions that takes into account every consequence that might be true in any total order, called *improbably weak projection* (Figure 4). Second, we provide a criterion for true throughout that succeeds only if the corresponding formula will be true in all total orders consistent with the current set of constraints. We define *improbably strong true throughout*:

$$
\begin{array}{l}
\forall \pi_1 \pi_2 \in \Pi \\
\quad \exists\, t_P \in \mathsf{T} \\
\qquad \mathit{STRONGLY\text{-}PROTECTED}(t_P) \wedge \\
\qquad (\mathrm{BEGIN}(t_P) \preceq \pi_1) \wedge \\
\qquad (\forall t_{\neg P} \in \mathsf{T} \\
\qquad\quad (\mathrm{STATUS}(t_{\neg P}) = \mathtt{OUT}) \vee \\
\qquad\quad (\mathrm{BEGIN}(t_{\neg P}) \prec \mathrm{BEGIN}(t_P)) \vee \\
\qquad\quad (\pi_2 \prec \mathrm{BEGIN}(t_{\neg P})) \ ) \\
\quad \Leftrightarrow \ \mathit{TT}(P, \pi_1, \pi_2)
\end{array}
\tag{4}
$$

There is a simple decision procedure for generating all consequences and computing the set of strongly protected tokens. Let $\mathsf{T}_1 = \mathsf{T}_B$, and initially assume that no tokens are strongly protected. Let $i = 1$. To compute the consequences of T_i, set $\mathsf{T}_{i+1} = \mathsf{T}_i$, compute the consequent tokens of each token in T_i using the criterion of improbably weak projection, and add any new tokens to T_{i+1}. Continue to compute new consequent tokens in this manner incrementing i as needed until $\mathsf{T}_i = \mathsf{T}_{i+1}$. Set $\mathsf{T} = \mathsf{T}_i$. At this point, perform a sweep forward in time (relative to the current partial order) determining for each token in T whether or not it is strongly protected and the status, IN or OUT, of each its consequents. In [6], we prove that this decision procedure is sound for a partially ordered set of tokens, and sound and complete for a totally ordered set.

In the same paper, we give two incremental update algorithms with polynomial-time worst case behavior, one for weak projection and weak true throughout, and one for improbably weak projection and improbably strong true throughout. The latter algorithm does not model the decision procedure given above—there is a more complicated procedure with the same behavior that is more efficient by a constant factor. We prove that these algorithms support exactly the conclusions licensed by their respective inference methods. Proving that the algorithms terminate is in one sense impossible. Using the TDBMS and a type 1 causal theory with arithmetic functions (e.g., `(project (contents ?register ?n) (increment ?register) (contents ?register (+ 1 ?n))))`, we can easily simulate a Turing machine. There are a number of methods for either restricting what the user can encode in a causal theory or limiting the scope of the inferences computed by the TDBMS in such a way that we can guarantee that the update terminates. By limiting the scope of the inferences computed by the TDBMS, we potentially sacrifice completeness, but we have shown that to ensure completeness may require an exponential amount for time for other reasons.

6 Conclusions

This paper is concerned with computational approaches to reasoning about time and causality, particularly in domains involving partial orders and incomplete information. We have described a class of causal theories

involving a carefully restricted use of nonmonotonic inference, capable of representing conditional effects and the effects of simultaneous actions. We showed in [6] that the decision problem for nontrivial inference systems involving conditional effects and partially ordered events is *NP-complete*. As an alternative to a complete but potentially exponential-time inference procedure, we have described a decision procedure, for which there is an incremental polynomial-time algorithm, that generates a useful subset of those inferences that will be true in all total orders consistent with the initially specified partial order. The decision procedure is provably sound and the resulting conclusions are guaranteed consistent if the underlying causal theory is consistent. If the events turn out to be totally ordered, the procedure is complete as well as sound.

References

1. Chapman, David, *Planning for Conjunctive Goals*, Technical Report AI-TR-802, MIT AI Laboratory, 1985.

2. Charniak, Eugene and McDermott, Drew V., *Introduction to Artificial Intelligence* (Addison-Wesley Publishing Co., 1985).

3. Dean, Thomas, *Temporal Imagery: An Approach to Reasoning about Time for Planning and Problem Solving*, Technical Report 433, Yale University Computer Science Department, 1985.

4. Dean, Thomas, and McDermott, Drew V., Temporal Data Base Management, to appear in *Artificial Intelligence*.

5. Dean, Thomas, An Approach to Reasoning About the Effects of Actions for Automated Planning Systems, to appear in the 1987 volume of the *Annals of Operations Research* entitled "Approaches to Intelligent Decision Support".

6. Dean, Thomas, and Boddy, Mark, *Incremental Causal Reasoning*, Technical Report CS-87-1, Brown University Department of Computer Science, 1987.

7. Doyle, Jon, A truth maintenance system, *Artificial Intelligence* **12** (1979) 231–272.

8. Fikes, Richard and Nilsson, Nils J., STRIPS: A new approach to the application of theorem proving to problem solving, *Artificial Intelligence* **2** (1971) 189–208.

9. Reiter, Raymond, A Logic for Default Reasoning, *Artificial Intelligence* **13** (1980).

10. Sacerdoti, Earl, *A Structure for Plans and Behavior* (American Elsevier Publishing Company, Inc., 1977).

11. Shoham, Yoav, *Chronological Ignorance: Time, Knowledge, Nonmonotonicity and Causation*, in Georgeff, Michael P. and Lansky, Amy L. (Eds.), *The 1986 Workshop on Reasoning about Actions and Plans*, (Morgan-Kaufman 1987).